Haskell

time limit per test: 1 second (3 seconds for Python)
memory limit per test: 500 megabytes
input: standard input
output: standard output
points: 100 · problem code: haskell

Indra is excited, because they have just learned Haskell! Now that they have acquired the skills of a functional programming god, they want to start working on a project made of n tasks. There are m ways to move between tasks while working: each move goes from some task a to another task b and changes Indra's **productivity** by an integer c. Positive values help; negative values represent fighting compiler errors.

Indra can start at any task and keep following moves, possibly using the same move more than once. Josh has claimed that no matter how Indra works, they can never return to a previously visited task with a strictly positive total productivity change along the way. In other words, Josh claims there is no productive loop in the project.

Help Indra check whether Josh is wrong.

A move from a task to itself counts as a loop. There may be several different moves between the same pair of tasks.

Input

The first line contains two integers n and m.

The next m lines each contain three integers a, b, and c, describing a move from task a to task b that changes productivity by c.

Tasks are numbered 1 through n.

Output

Print `YES` if there exists a sequence of moves that starts and ends at the same task and has strictly positive total productivity change. Otherwise, print `NO`.

Constraints

- $1 \le n \le 1000$
- $0 \le m \le 10000$
- $1 \le a, b \le n$
- $-100 \le c \le 100$

Example 1

Input

```
3 6
3 1 -80
1 3 -58
1 3 81
2 3 -81
2 1 24
2 1 -55
```

Output

```
YES
```

Explanation

Starting at task 1, Indra can move $1 \to 3$ with productivity 81, then $3 \to 1$ with productivity -80, returning to task 1 with total change 1.

Example 2

Input

```
2 2
1 2 10
2 1 -10
```

Output

```
NO
```

Explanation

The only way to return to the starting task has total productivity change 0, which is not strictly positive.